UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE
INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TRABAJO ENCARGADO

MÚLTIPLES CAPAS

CURSO:

ESTRUCTURAS DE DATOS AVANZADOS

DOCENTE:

ING. FREDY COLLANQUI MARTÍNEZ

PRESENTADO POR:

BELTRAN EDWIN MAMANI MAMANI

PUNO- PERU

2024

MÚLTIPLES CAPAS (MULTIPLESLAYERS)

SEMANA 16 (JULIO 15, 17 ,19)

El uso de múltiples capas en un mapa permite organizar y visualizar diferentes tipos de datos espaciales de manera eficiente. Este enfoque facilita el análisis y la toma de decisiones en aplicaciones GIS, mostrando cómo interactúan distintos elementos espaciales entre sí.

EJEMPLO PRÁCTICO

Vamos a usar datos reales para mostrar cómo las "Múltiples capas" pueden ser aplicadas en un sistema de información geográfica (GIS) utilizando la biblioteca folium en Python. En este ejemplo, utilizaremos datos públicos disponibles sobre la ciudad de Nueva York, como estaciones de metro y parques.

Estaciones de Metro: Puntos que representan las ubicaciones de las estaciones de metro.

Parques: Polígonos que representan las áreas de los parques.

MAPA DE NUEVA YORK CON CAPAS UTILIZANDO LEAFLET

```
<html>
<head>
  <title> Mapa de Nueva York con capas </title>
  <meta  charset= "utf-8"  />
  <meta  name= "viewport"  content= "width=device-width, initial-scale=1.0" >
  <link  rel= "stylesheet"  href= "https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"  />
  <script src= "https://unpkg.com/leaflet@1.7.1/dist/leaflet.js" ></script>
</head>
<body>
```

- <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />: Incluye la hoja de estilos de Leaflet.
- <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>: Incluye la biblioteca JavaScript de Leaflet

```
<body>
  <div  id= "mapa"  style= "ancho: 100%; alto: 600px;" ></div>
```

- <body>: Contiene el contenido visible de la página web.
- : Define un contenedor div para el mapa con un identificador id="map" y un estilo que especifica su tamaño.

Script JavaScript

```
< script >
```

```
2       // Inicializar el mapa centrado en Nueva York
3       var map = L.map( 'map' ).setView([ 40.7128 , - 74.0060 ], 12 );
```

var map = L.map('map').setView([40.7128, -74.0060], 12);: Inicializa el mapa centrado en las coordenadas de Nueva York (latitud 40.7128, longitud -74.0060) con un nivel de zoom 12.

```
                // Datos GeoJSON de estaciones de metro
1               var undergroundData = {
2                   "type" :  "FeatureCollection" ,
3                   "features" : [
4                     {
5                       "tipo" :  "Característica" ,
6                       "geometría" : {
7                         "tipo" :  "Punto" ,
8                         "coordenadas" : [ - 74.005941 , 40.712784 ]
9                       },
10                      "propiedades" : {
11                        "nombre" :  "Estación 1"
12                      }
13                    },
14                    {
15                      "tipo" :  "Característica" ,
dieciséis               "geometría" : {
17                        "tipo" :  "Punto" ,
18                        "coordenadas" : [ - 73.985428 , 40.748817 ]
19                      },
20                      "propiedades" : {
21                        "nombre" :  "Estación 2"
22                      }
23                    }
24                    // Más datos...
25                ]
26              };
27
```

var subwayData = { ... };: Define un objeto GeoJSON que contiene los datos de las estaciones de metro. Cada estación es una característica (Feature) con geometría de tipo Point y propiedades como el name

```
1               // Datos GeoJSON de parques
2               var parksData = {
3                   "type" :  "FeatureCollection" ,
4                   "features" : [
5                     {
6                       "tipo" :  "Característica" ,
7                       "geometría" : {
8                         "tipo" :  "Polígono" ,
9                         "coordenadas" : [[
10                          [ - 74.012084 , 40.705569 ],
11                          [ - 74.013642 , 40.706225 ],
```

```
                        [ -74.013281 , 40.707119 ],
                        [ -74.011751 , 40.706493 ],
                        [ -74.012084 , 40.705569 ]
                    ]]
                },
                "propiedades" : {
                    "nombre" :  "Parque 1"
                }
            },
            {
                "tipo" :  "Característica" ,
                "geometría" : {
                    "tipo" :  "Polígono" ,
                    "coordenadas" : [[
                        [ -73.968285 , 40.785091 ],
                        [ -73.958285 , 40.785091 ],
                        [ -73.958285 , 40.795091 ],
                        [ -73.968285 , 40.795091 ],
                        [ -73.968285 , 40.785091 ]
                    ]]
                },
                "propiedades" : {
                    "nombre" :  "Parque 2"
                }
            }
            // Más datos...
        ]
    };
```

var parksData = { ... };: Define un objeto GeoJSON que contiene los datos de los parques. Cada parque es una característica (Feature) con geometría de tipo Polygon y propiedades como el name.

```
// Estilo para las estaciones de metro
function metroStyle(feature) {
    return {
        radio :  8 ,
        fillColor :  "#0000FF" , //
        Color azul :  "#000" ,
        peso :  1 ,
        opacidad :  1 ,
        Opacidad de relleno :  0,8
    };
}
```

function subwayStyle(feature) { ... };: Define una función que devuelve un objeto de estilo para las estaciones de metro. Las estaciones se representan como marcadores circulares (circleMarker) de color azul.

```
// Estilo para los parques
function parksStyle(feature) {
    return {
      color :  "#00FF00" , // Verde
      peso :  2 ,
      opacidad :  1 ,
      Opacidad de relleno :  0,2
    };
}
```

function parksStyle(feature) { ... };: Define una función que devuelve un objeto de estilo para los parques. Los parques se representan como polígonos de color verde.

```
// Añadir capa de estaciones de metro
var metroLayer = L.geoJson(subwayData, {
    pointToLayer :  función (característica, latitud) {
        return L.circleMarker(longitud, metroStyle(característica));
    },
    onEachFeature :  función (característica, capa) {
        si (característica.propiedades && característica.propiedades.nombre) {
          capa.bindPopup(caracteristica.propiedades.nombre);
        }
    }
});


metroCapa.addTo(mapa);
```

var subwayLayer = L.geoJson(subwayData, { ... });: Crea una capa GeoJSON para las estaciones de metro.

- pointToLayer: function (feature, latlng) { ... };: Convierte los puntos en marcadores circulares aplicando el estilo definido en subwayStyle.
- onEachFeature: function (feature, layer) { ... };: Asocia un popup con el nombre de la estación a cada marcador.

```
// Añadir capa de parques
var parksLayer = L.geoJson(parksData, {
    estilo : parquesEstilo,
    onEachFeature :  función (característica, capa) {
        si (característica.propiedades && característica.propiedades.nombre) {
          capa.bindPopup(caracteristica.propiedades.nombre);
        }
    }
});
parksLayer.addTo(mapa);
```

var parksLayer = L.geoJson(parksData, { ... });: Crea una capa GeoJSON para los parques.

- style: parksStyle: Aplica el estilo definido en parksStyle a los polígonos de los parques.
- onEachFeature: function (feature, layer) { ... };: Asocia un popup con el nombre del parque a cada polígono.

Este código crea un mapa interactivo centrado en Nueva York, mostrando estaciones de metro y parques con estilos visuales distintos y controles de capas.

Utilizando la biblioteca Leaflet, se proporciona una experiencia de usuario enriquecida y personalizable para visualizar datos geoespaciales.